Internet Engineering Task Force
INTERNET-DRAFT draft-ietf-telnet-tls-00
expires July 1998

Michael Boe
Cisco Systems

December, 1997

TLS-based Telnet Security

Status of this memo

Abstract

Telnet service has long been a standard Internet protocol. However, a standard way of ensuring privacy and integrity of Telnet sessions has been lacking. This document proposes a standard method for Telnet servers and clients to use the Transport Layer Security (TLS) protocol. It describes how two Telnet participants can decide whether or not to attempt TLS negotiation, and how the two participants should process authentication credentials exchanged as a part of TLS startup.

Changes since last Draft

The first draft was really just writing down what's been done in one working Telnet implementation where TLS is used: the SSLeay SSLTelnet package. This seems as good a starting point as any. Reviewers had the following comments concerning the content of that draft:

*separate port: Why not just do a separate port implementation?

At the Washington D.C. meeting, it was made clear that specifying a separate port implementation would be met with a distinct lack of enthusiasm on the part of the IESG. So the separate port is still absent.

*no RFC1416: Why do the RFC1416 song and dance when the RFC1416 model doesn't fit with this?

John Hind points out that one could want to negotiate Kerberos on top of TLS. Chris Newman takes this a step further and points out that TLS is not really doing the same thing as RFC1416; TLS is providing transport encryption and optional authentication; RFC1416 appears to be providing authentication and sometimes encryption. I've scuttled the use of RFC1416 to negotiate TLS. Instead, TLS is negotiated via a IAC DO STARTTLS option. This breaks SSLeay implementations.

*0xff processing: We may need to explicitly disable Telnet 0xff processing during TLS negotiation.

I'm not sure what to think of this. My hope is that by accepting the IAC DO STARTTLS option, normal Telnet 0xff processing is suspended until TLS negotiation has completed (successfully or not). If someone think this won't work (due to lack of synchronicity at the end TLS negotiations), please let me know.

Intent of Draft

I'm hoping this draft provides some focus on the following issues:

- how client and server can achieve TLS/SSL-based Telnet connections.

- how TLS can co-exist on the same server (or client) program with other authentication techniques. This is an interesting problem, since it turns out I can't talk about some of these techniques

because RFC1416 is Experimental, and normative references need to be Standards. So I'm having to hack the text to refer to these techniques generically and not specifically. I may be able to get away with using RFC1416 as an example but pointing that it's just that: an example.

- the limitations (futility?) of attempting to ``negotiate'' TLS at the Telnet level.

I did not include descriptive narrative showing how things could be implemented. Although this is a ``good'' thing in general with RFCs, it makes initial understanding a bit harder. So please question the draft if something appears pointless or obscure.

One thing I have not addressed is the minimum set of cipher-suites that must be supported by all clients and servers. The IETF looks like it is settling on TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA, which is the suite of DSS certificates and triple-DES and SHA MAC. This is not what comes standard with Netscape SSL 3.0...but then, supposedly SSL is just a fallback in case one of the implementations is not compliant with this spec :-).

The reference to SASL in this draft is purely placeholder for future work. I don't envision this SASL ``protocol exchange'' that will provide authorization identity as being very complicated. I'm thinking it will be very simple--maybe a pair of exchanges between client and server sent immediately after the TLS negotation.

1 Introduction

We are interested in addressing the interaction between the Telnet client and server that will support this secure requirement with the knowledge that this is only a portion of the total end-user to application path. Specifically it is often true that the Telnet server does not reside on the target machine (it does not have access to a list of identities which are allowed to access to that mainframe), and it is often true (e.g. 3270 access) that the TN server can not even identify that portion of the emulation stream which contains user identification/password information. Additionally it may be the case that the Telnet client is not co-resident with the end user and that it also may be unable to identify that portion of the data stream that deals

with user identity. We make the assumption here that there is a trust relationship and appropriate protection to support that relationship between the TN Server and the ultimate application engine such that data on this path is protected and further that the application will authenticate the end user via the emulation stream as well as use this to

control access to information. We further make the assumption that the path between the end user and the client is protected.

To hold up our Telnet part of the overall secure path between the user and the mainframe we need to make the Telnet data stream unobservable to a third party. We do this by creating a shared secret between the client and server which is used to encrypt the flow of data and (just as important) require the client to verify that it is talking to correct server (the one that the mainframe trusts rather than an unintended man-in-the-middle) with the knowledge that the emulation stream itself will be used by the mainframe to verify the identity of the end-user. Rather than create a specialized new protocol which accomplishes these goals we instead have chosen to use existing protocols with certain constraints.

The TLS protocol (formerly known as SSL) can shield a connection from tampering and eavesdropping. One protocol which can benefit from the security TLS offers is Telnet [TELNET]. Although other security mechanisms have been used with Telnet (e.g. KERBEROS [RFC1416]), TLS offers a broad range of security levels that allow sites to proceed at an "evolutionary" pace in deploying authentication, authorization and confidentiality policies, databases and distribution methods.

TLS is used to provide the following:

- creation and refresh of a shared secret;
- negotiation and execution of data encryption and optional compressesion;
- primary negotiation of authentication; and, if chosen
- execution of public-key or symmetric-key based authentication.

Since both encryption and authentication is always needed for a secure channel that meets the requirements of these emulation types, we can use the anonymous authentication negotiation option of TLS as an indication that the client wants to negotiate some non-TLS-based authentication exchange. Note that as per usual TLS rules, the client must always authenticate the server's credentials.

TLS at most offers only authentication of the peers conducting the TLS dialog. In particular, it does not offer the possibility of the client providing separate credentials for authorization than were presented for authentication. It is expect that other RFCs will be produced describing how other authentication mechanisms can be used in conjunction with TLS.

Traditional Telnet servers have operated without such early presentation of authorization credentials for many reasons (most of which are historical). However, recent developments in Telnet server technology make it advantageous for the Telnet server to know the authorized capabilities of the remote client before choosing a communications link (be it `pty' or SNA LU) and link-characteristics to the host system (be that ``upstream'' link local or remote to the server). Thus, we expect to see the use of client authorization to become an important element of the telnet evolution. Such authorization methods may require certificates presented by the client via TLS, or by the use of SASL or RFC1416, or some other as yet unknown method.

This document defines extensions to Telnet which allow TLS to be activated early in the lifetime of the Telnet connection. It defines a set of advisory security-policy response codes for use when negotiating TLS over Telnet.

2 Conventions Used in this Document

The key words "REQUIRED", "MUST", "MUST NOT", "SHOULD", "SHOULD NOT" and "MAY" in this document are to be interpreted as described in [KEYWORDS].

Formal syntax is defined using ABNF [ANBF].

In examples, "C:" and "S:" indicate lines sent by the the client and server, respectively.

3 Telnet STARTTLS Option

The STARTTLS option is an asymmetric option, with only the server side being able to send IAC DO STARTTLS. The client may initiate by sending IAC WILL STARTTLS. There are some more rules due to the need to clear the link of data (or to ``synchronize'' the link). This synchronization takes the form of a three-way handshake:

1. As per normal Telnet option processing rules, the client MUST

respond to the server's IAC DO STARTTLS with either IAC WONT STARTTLS or IAC WILL STARTTLS (if it hasn't already done so). An affirmative response MUST be followed eventually by IAC SB STARTTLS FOLLOWS IAC SE. If the client sends the affirmative response, it must then not initiate any further Telnet options or subnegotiations except for the STARTTLS subnegotiation until after the TLS

negotiation has completed.

2. The server SHOULD NOT send any more Telnet data or commands after sending IAC DO STARTTLS except in response to client Telnet options received until after it receives either a negative response from the client (IAC WONT STARTTLS) or the affirmative (both IAC WILL STARTTLS and followed eventually by IAC SB STARTTLS FOLLOWS IAC SE). If the client's STARTTLS option response is negative, the server is free to again send Telnet data or commands. If the client's response is affirmative, then the server MUST send only IAC SB STARTTLS FOLLOWS IAC SE. If the server sends IAC SB STARTTLS FOLLOWS IAC SE, then the server MUST also arrange at this time, for the normal Telnet byte-stuff/destuff processing to be turned off for the duration of the TLS negotiation.

3. The client, upon receipt of the server's IAC SB STARTTLS FOLLOWS IAC SE, MUST also arrange for normal Telnet byte-stuff/destuff processing to be turned off for the duration of the TLS negotiation. It MUST then enter into TLS negotiation by sending the TLS ClientHello message.

Here's a breakdown of the Telnet command phrases defined in this document:

S: IAC DO STARTTLS-- Sent only by server. Indicates that server strongly desires that the client enter into TLS negotiations.

C: IAC WILL STARTTLS-- Sent only by client. Indicates that client strongly desires that the server enter into TLS negotiations.

S: IAC DONT STARTTLS-- Sent only by the server. Indicates that the server is not willing to enter into TLS negotations.

C: IAC WONT STARTTLS-- Sent only by the client. Indicates that the client is not willing to enter into TLS negotiations.

C: IAC SB STARTTLS FOLLOWS IAC SE-- When sent by the client, this indicates that the client is preparing for TLS negotiations, and that the next thing sent by the client will be the TLS ClientHello.

S: IAC SB STARTTLS FOLLOWS IAC SE-- When sent by the server, this indicates that the server has prepared to receive the TLS ClientHello from the client.

3.1 Abnormal Negotiation Failure

The behavior regarding TLS negotiation failure is covered in [TLS]. The TLS spec recommends that the TCP connection be dropped where negotiation failure is due to insufficient security, inability to verify a certificate, handshake failure, etc. This appears not to indicate that the TCP connection be broken; the semantics are that TLS is finished and all state variables cleaned up.

If this happens, the two sides may continue the Telnet connection. Here's how:

- The side which received the TLS ErrorAlert message speaks first;
- The first speaker can send any Telnet data or commands;
- Neither side SHOULD attempt to issue another STARTTLS during the lifetime of the Telnet session.

4 TLS, Authentication and Authorization

If TLS has been successfully negotiated, the client will have the server's certificate. This indicates that the server's identity can be verified. Client implementations MUST always verify the server identity as part of TLS processing and fail the connection if such verification fails. See Section /refsec:servauth for details of the mechanics of server authentication.

The server, however, may not have the client's identity (verified or not). This is because the client need not provide a certificate during the TLS exchange. Or it may be server site policy not to use the identity so provided. In any case, the server may not have enough confidence in the client to move the connection to the ``authenticated'' state.

On the other hand, the server MAY elect to use the client's TLS-proffered credentials as the basis for authentication. If the server is satisfied

with the credentials, it MUST move the connection to the ``authenticated'' state before it processes any further Telnet data or commands from the client.

Once the connection has been moved to the authenticated state, the server MUST NOT initiate further authentication-related Telnet exchanges with the client after moving the Telnet connection to the

authenticated state. And the server MUST NOT accept further attempts by the client to proffer authentication details.

[Author's note: this has some interesting implications. It appears that it's very important for the client to pick a TLS authentication mechanism that has the best chance in resulting in the authorizations wanted by that client. That's because the server will very probably prefer TLS-provided authentication over post-TLS-negotiation authentication. Are people happy with this?]

If further client, server or client-server authentication is going to occur after TLS has been negotiated, it MUST occur before any non-authentication-related Telnet interactions take place on the link after TLS starts. This specification recognizes Telnet AUTHENTICATE and TUID options and subsequent subnegotiations as being valid authentication-related Telnet interactions (for more information on these to options, see [RFC1416] and [RFC927] respectively). [Note that this specification does not rely on TUID or AUTHENTICATE Telnet options in any way.] Further Telnet options and subnegotiations may be added to this list by registering them with IANA as being authentication-related Telnet options. When the first non-authentication-related Telnet interaction is received by either participant, then the receiving participant MAY drop the connection due to dissatisfaction with the level of authentication.

[author's note: Probably a few error-conditions ala ACAP/POP3/IMAP authentication are in order so that the spurned participant has something useful to display/log other than "session disconnected." Comments?]

And no TLS negotiation outcome, however trustworthy, will by itself provide the server with the authorization identity if that is different from the authentication identity of the client. See [SASL] for why this might be a desirable function to have with proxy clients, etc. How such authorization is done is outside the scope of this document.

The following subsections detail how the client can provide the server with authentication and authorization credentials separate to the TLS mechanism.

4.1 PKI-based Authentication and certificate extensions

When PKI authentication is used no special X.509 certificate extensions will be require but a client or server may choose to support an extension if found. For example, they may use a contained certificate revocation

URL extension if provided. The intent is to allow sharing of certificates with other services on the same host, for example, a Telnet server might use the same certificate to identify itself that a co-located WEB server uses. The method of sharing certificates is outside the scope of this document.

4.2 Pre-TLS Authentication

It could be that the server had moved the Telnet connection to the authenticated state sometime previous to the negotiation of TLS. If this is the case, then the server MUST NOT use the credentials proffered by the client during the TLS negotiations for authorization of that client. The server should, of course, verify the client's TLS-proffered credentials.

[Author's note: I've deleted the sections on RFC1416 and SASL. RFC1416 is an Experimental RFC, and I don't want people to think that this document relies on an experimental RFC (which would prevent this document from being approved). And the use of SASL should be a completely separate effort from the one this document describes.]

5 Security

Security is discussed throughout this document. Most of this document concerns itself with wire protocols and security frameworks. But in this section, client and server implementation security issues are in focus.

5.1 Authentication of the Server by the Client

How the client can verify the server's proferred identity varies according to the key-exchange algorithm used in the selected TLS cipher-suite. However, it's important for the client to follow good security practice in verifying the proffered identity of the server.

5.1.1 PKI-based certificate processing

When PKI authentication is used no special X.509 certificate extensions will be required but a client or server may choose to support an extension if found. For example, they may use a contained certificate revocation URL extension if provided. The intent is to allow sharing of

certificates with other services on the same host, for example, a Telnet server might use the same certificate to identify itself that a co-located WEB server uses. The method of sharing certificates is not a topic of this document.

The verification of the server's certificate by the client MUST include, but isn't limited to, the verification of the signature certificate chain to the point where the a signature in that chain uses a known good signing certificate in the clients local key chain. The verification SHOULD then continue with a check to see if the fully qualified host name which the client connected to appears anywhere in the server's certificate subject (DN). If no match is found then the end user should see a display of the servers certificate and be asked if he/she is willing to proceed with the session.

[question: do certs always get signed with the DNS domain-name? Further, will all clients that implement TLS also be forced to use DNS? If not, how would the client end up knowing the DNS name of the host? Another question: suppose the FQDN of the server comes back as a subdomain of the name on the cert. Is this ok? What are the pros and cons of this?]

5.1.2 Kerberos V5 server verification

[Nothing here yet. Just a placeholder to show everybody that PKI-based authentication is not the only game in town.]

5.2 Display of security levels

The Telnet client and server MAY, during the TLS protocol negotiation phase, choose to use a weak cipher suite due to policy, law or even convenience. It is, however, important that the choice of weak cipher suite be noted as being commonly known to be vulnerable to attack. In particular, both server and client software should note the choice of weak cipher-suites in the following ways:

- If the Telnet endpoint is communicating with a human end-user, the user-interface SHOULD display the choice of weak cipher-suite and

the fact that such a cipher-suite may compromise security.

- The Telnet endpoints SHOULD log the exact choice of cipher-suite as part of whatever logging/accounting mechanism normally used.

6 TLS Variants and Options

TLS has different versions and different cipher suites that can be supported by client or server implementations. The following subsections detail what TLS extensions and options are mandatory. The subsections also address how TLS variations can be accommodated.

6.1 Support of previous versions of TLS

TLS has its roots in SSL 2.0 and SSL 3.0. Server and client implementations may wish to support for SSL 3.0 as a fallback in case TLS **3.1 or higher is not supported. This is permissible; however, client** implementations which negotiate SSL3.0 MUST still follow the rules in Section 5.2 concerning disclosure to the end-user of transport-level security characteristics.

Negotiating the use of SSL 3.0 is done as part of the TLS negotiation; it is detailed in [TLS]. Negotiating SSL 2.0 is not recommended.

6.2 Using Kerberos V5 with TLS

If the client and server are both amenable to using Kerberos V5, then using non-PKI authentication techniques within the confines of TLS may be acceptable (see [TLSKERB]). Note that clients and servers are under no obligation to support anything but the cipher-suite(s) mandated in [TLS]. However, if implementations do implement the KRB5 authentication as a part of TLS ciphersuite, then these implementations SHOULD support at least the TLS_KRB5_WITH_3DES_EDE_CBC_SHA ciphersuite.

7 Protocol Examples

The following sections provide skeletal examples of how Telnet clients and servers can negotiate TLS.

7.1 Successful TLS negotiation

The following protocol exchange is the typical sequence that starts TLS:

```
// typical successful opening exchange
```

```
  S: IAC DO STARTTLS
  C: IAC WILL STARTTLS IAC SB STARTTLS FOLLOWS IAC SE
  S: IAC SB STARTTLS FOLLOWS IAC SE
// server now readies input stream for non-Telnet, TLS-level negotiation
  C: [starts TLS-level negotiations with a ClientHello]
  [TLS transport-level negotiation ensues]
  [TLS transport-level negotiation completes with a Finished exchanged]
// either side now able to send further Telnet data or commands
```

The following protocol exchange is the typical sequence that starts TLS, but with the twist that the (TN3270E) server is willing but not aggressive about doing TLS; the client strongly desires doing TLS.

```
// typical successful opening exchange
  S: IAC DO TN3270E
  C: IAC WILL STARTTLS IAC
  C: IAC WONT TN3270E
  S: IAC DO STARTTLS
  C: IAC WILL STARTTLS IAC SB STARTTLS FOLLOWS IAC SE
  S: IAC SB STARTTLS FOLLOWS IAC SE
// server now readies input stream for non-Telnet, TLS-level negotiation
  C: [starts TLS-level negotiations with a ClientHello]
  [TLS transport-level negotiation ensues]
  [TLS transport-level negotiation completes with a Finished
                exchanged]
// note that server retries negotiation of TN3270E after TLS
// is done.
  S: IAC DO TN3270E
  C: IAC WILL TN3270E
// TN3270E dialog continues....
```

7.2 Unsuccessful TLS negotiation

This example assumes that the server does not wish to allow the Telnet session to procede without TLS security; however, the client's version of TLS does not interoperate with server's.

```
//typical unsuccessful opening exchange
```

```
  S: IAC DO STARTTLS
  C: IAC WILL STARTTLS IAC SB STARTTLS FOLLOWS IAC SE
  S: IAC SB STARTTLS FOLLOWS IAC SE
// server now readies input stream for non-Telnet, TLS-level negotiation
  C: [starts TLS-level negotiations with a ClientHello]
```

```
  [TLS transport-level negotiation ensues]
  [TLS transport-level negotiation fails with server sending
              ErrorAlert message]
  C: IAC DO TERMINAL-TYPE
  S: IAC WONT TERMINAL-TYPE
  S: [TCP level disconnect]
//  server (or both) initiate TCP session disconnection
```

This example assumes that the server wants to do TLS, but is willing to allow the session to procede without TLS security; however, the client's version of TLS does not interoperate with server's.

```
//typical unsuccessful opening exchange
  S: IAC DO STARTTLS
  C: IAC WILL STARTTLS IAC SB STARTTLS FOLLOWS IAC SE
  S: IAC SB STARTTLS FOLLOWS IAC SE
// server now readies input stream for non-Telnet, TLS-level negotiation
  C: [starts TLS-level negotiations with a ClientHello]
  [TLS transport-level negotiation ensues]
  [TLS transport-level negotiation fails with server sending
              ErrorAlert message]
  C: IAC DO TERMINAL-TYPE
  S: IAC WILL TERMINAL-TYPE
// regular Telnet dialog ensues
```

References

[ANBF] D. Crocker, Ed., P. Overell, ``Augmented BNF for Syntax Specifications: ABNF'', RFC2235, November 1997.

[KEYWORDS] Bradner, S. ``Key words for use in RFCs to Indicate Requirement Levels'', RFC2119, March 1997.

[RFC927] Brian A. Anderson. ``TACACS User Identification Telnet Option'', RFC927, December 1984

[RFC1416] D. Borman, Editor. ``Telnet Authentication Option'', RFC1416, February 1993.

[SASL] Myers, J. ``Simple Authentication and Security Layer

(SASL)'', RFC2222, October 1997.

[TELNET] J. Postel, J. Reynolds. ``Telnet Protocol Specifications'', RFC854, May 1983.

[TLS] Tim Dierks. ``The TLS Protocol'', Internet Draft, November 1997.

[TLSKERB] Ari Medvinsky, Matthew Hur. ``Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)'', Internet Draft, July 1997.

Author's Address

Michael Boe
Cisco Systems Inc.
1264 5th Avenue
San Francisco, CA 94122-2649

Email: Michael Boe <mboe@cisco.com>